
Politecnico di Torino

Digital systems electronics
A.A. 2018/2019

Prof. G. Masera

---

# Lab 07

**21 may 2019**

---


| | |
|---|---|
| Berchialla Luca | 236032 |
| Laurasi Gjergji | 238259 |
| Mattei Andrea | 233755 |
| Lombardo Domenico Maria | 233959 |
| Wylezek Karolina | 267219 |

## 0.1 Generating a square wave

In order to make this part of the laboratory we modified the code from the previous exercise. We changed the output pin from the one associated with the LED to the one related to the D2 pin of the board (PA10 of the microcontroller). At the beginning we decided to try the code without adding the code given in the description of exercise and we saw a stable waveform. After adding the code, we noticed that the period of the square wave was unstable. The reason for this unstability is the fact that the additional code generated a pseudo-random delays every system clock tick.

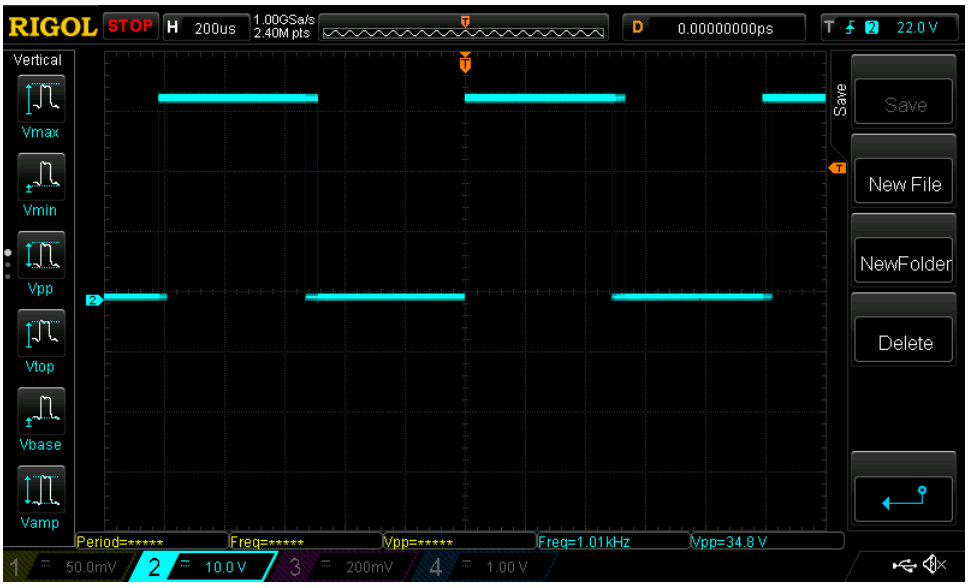


Figure 1: the generated waveform with the disturbance

# 1.1 Using a switch to toggle the LED, Low Level Approach

Following the instructions in the Lab7document we created an empty project in SWB4STM32, than the pointers for the GPIOA_MODER, GPIOA_ODR, GPIOC_MOER, GPIOC_IDR and RCC_AHB1ENR have been initialized.
The clock has been configured to 1 for the GPIOA and GPIOC devices, after that the GPIOA pin5 has been configured as output and GPIOC pin 13 as input.
The application code starts with the initialization of the led to $off$ than we acquire a first time the complemented value of the button in button_old. In the infinite while loop as first time we acquire the complemented button's value in button, after we if a 0-1 transition happened we complement the bit 5 of GPIO_ODR. As last action we update button_old with the value button.

## 1.2 Using a switch to toggle the LED, CubeMX approach

In this section we'll deal with the same exercise as the previous one, using a CubeMX approach with LL (Low Layer) libraries.

In CubeMX we set the GPIO input and output respectively to pin A5 and C13 to correctly configure the LED and Button.

Apart from the I/O configuration, the algorithm is implemented as the point 1.1.

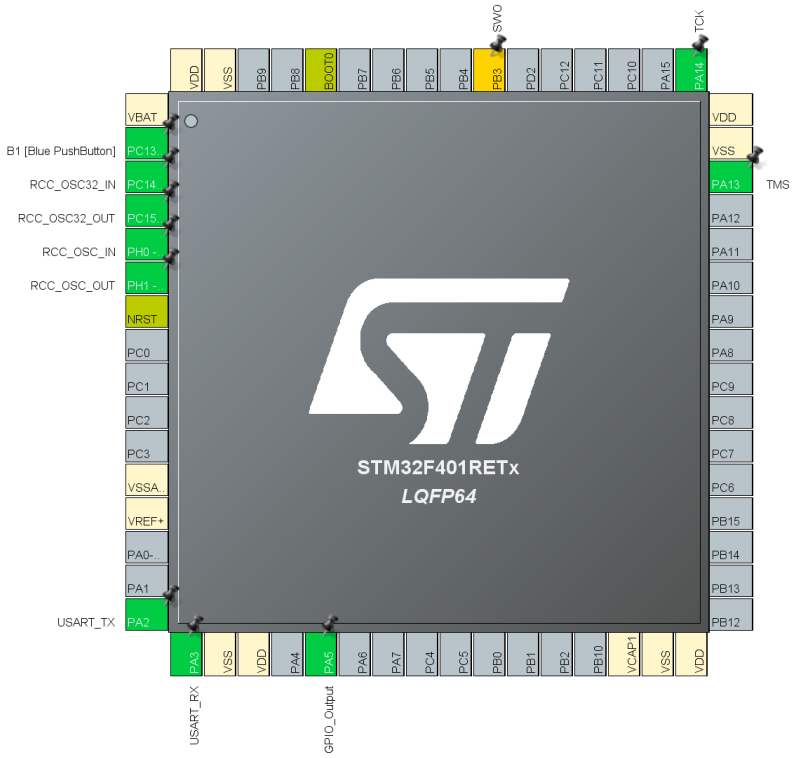


Figure 2: CubeMX GPIO configurations

# 2 Varying the blinking frequency

Varying the maximum "for" counting number we can play on the blinking frequency. The maximum appreciable frequency is around , due to 2 main reasons: The maximal frame rate humans can perceive is around 70 Hz. Furthermore the LED being not an ideal component shows parasitic effects (Equivalent RC filter) that decrease the maximum appreciable frequency to about 60 Hz.